Laboratorium Programowania Komputerów 4

Ćwiczenie nr 6 - inteligentne wskaźniki

Wstęp: utworzyć pusty projekt (empty project) i dodać do niego wszystkie pliki dołączone do zadania. Należy modyfikować wyłącznie wskazane części plików *zadanie1.cpp*, *zadanie2.cpp* oraz *zadanie3.cpp*. Nie należy modyfikować pozostałych plików. Przed rozpoczęciem rozwiązywania zadań, program powinien dać się skompilować i uruchomić. Program wypisuje na ekran ilość obiektów pewnego typu które zostały zaalokowane, a nie zwolnione, a więc swego rodzaju miarę wycieku pamięci. Po poprawnym rozwiązaniu zadania powinna być wypisywana wartość zero.
Uwaga: usunięcie wycieku pamięci nie może odbywać się kosztem funkcjonalności programu (jeżeli w liście nie będziemy alokować pamięci, to nie będzie wycieku pamięci, jednak lista przestanie działać – nie jest to akceptowalne rozwiązanie).

Zadanie 1. `unique_ptr`

Zadanie składa się z dwóch części i polega na usunięciu wycieków pamięci z otrzymanej funkcji na dwa sposoby – z użyciem inteligentnych wskaźników i bez nich.

a) zmodyfikuj funkcję `zadanie1a()` znajdującą się w pliku *zadanie1.cpp* tak, by usunąć wycieki pamięci. Nie używaj inteligentnych wskaźników.
b) zmodyfikuj funkcję `zadanie1b()` znajdującą się w pliku *zadanie1.cpp* tak, by usunąć wycieki pamięci. Użyj wskaźnika `std::unique_ptr`. Nie używaj operatorów `new` i `delete`. Porównaj funkcję `zadanie1a()` oraz `zadanie1b()` pod kątem długości kodu, złożoności oraz łatwości dalszej rozbudowy.
c) zmodyfikuj funkcję `checkSomething2()` znajdującą się w pliku *zadanie2.cpp* tak, aby możliwe było zastąpienie funkcji `checkSomething()` w instrukcji warunkowej w `case 2:` funkcji `zadanie1b()` wywołaniem funkcji `checkSomething2()` w postaci `checkSomething2(data)`.

Zadanie 2. `shared_ptr`

Zadanie składa się z dwóch części i polega na usunięciu wycieków pamięci z danej klasy będącej implementacją listy jednokierunkowej na dwa sposoby – z użyciem inteligentnych wskaźników i bez nich.

a) zmodyfikuj klasę `LinkedList_a` znajdującą się w pliku *zadanie2.cpp* tak, aby usunąć wycieki pamięci. Nie używaj inteligentnych wskaźników. Konieczna jest modyfikacja jedynie metod usuwających `pop_front()`, `erase()` i `clear()`.
b) zmodyfikuj klasę `LinkedList_b` znajdującą się w pliku *zadanie2.cpp* tak, aby usunąć wycieki pamięci. Użyj wskaźnika `std::shared_ptr`. Nie używaj operatorów `new` i `delete`. Porównaj klasę `LinkedList_a` oraz `LinkedList_b`. Jakie są wady klasy `LinkedList_b` względem klasy `LinkedList_a`?

Uwaga: w pliku *zadanie2.cpp* znajdują się funkcje `zadanie2a()` oraz `zadanie2b()`. Nie należy ich modyfikować.
Uwaga: w pliku *zadanie2.cpp* znajduje się zakomentowana definicja preprocesora `#define DEBUG`. Odkomentowanie jej powoduje że podczas na ekranie wypisywana jest zawartość listy po wykonaniu na niej operacji. Może być to przydatne narzędzie w razie podejrzewania błędnego działania listy, jednak do prezentacji wyników definicja ta powinna być zakomentowana.

Zadanie 3. `weak_ptr`

Zadanie polega na modyfikacji klasy będącej implementacją listy dwukierunkowej tak, aby korzystała z inteligentnych wskaźników.

a) zmodyfikuj klasę `DoublyLinkedList` znajdującą się w pliku *zadanie3.cpp* tak, aby korzystała z inteligentnych wskaźników. Nie używaj operatorów `new` i `delete`.

Uwaga: w pliku *zadanie3.cpp* znajduje się funkcja `zadanie3()`. Nie należy jej modyfikować.
Uwaga: w pliku *zadanie3.cpp* znajduje się zakomentowana definicja preprocesora `#define DEBUG`. Odkomentowanie jej powoduje że podczas na ekranie wypisywana jest zawartość listy po wykonaniu na niej operacji. Może być to przydatne narzędzie w razie podejrzewania błędnego działania listy, jednak do prezentacji wyników definicja ta powinna być zakomentowana.

Przydatne linki:
http://www.cplusplus.com/reference/memory/unique_ptr/
http://www.cplusplus.com/reference/memory/shared_ptr/
http://www.cplusplus.com/reference/memory/weak_ptr/